Technical Report

CMU/SEI-93-TR-08
ESC-TR-93-185

Software Engineering Institute

A Case Study in Software Maintenance

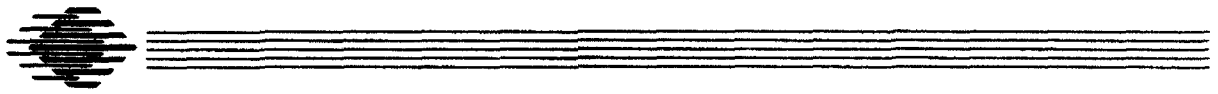
Susan Dart
Alan M. Christie
Alan W. Brown

July 1993

Technical Report

CMU/SEI-93-TR-08
ESC-TR-93-185
July 1993

A Case Study in Software Maintenance



Susan Dart
Alan M. Christie
Alan W. Brown

Case Environments Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

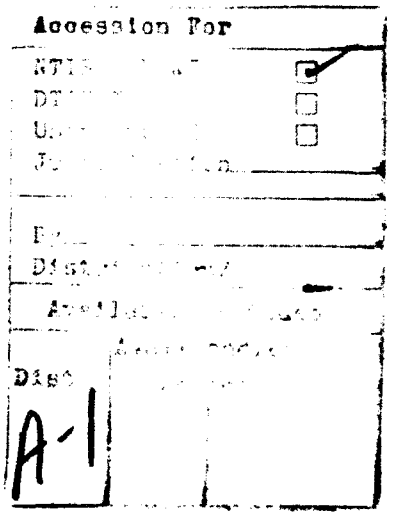
This technical report was prepared for the

SEI Joint Program Office
ESC/ENS
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

Review and Approval

This report has been reviewed and is approved for publication.



FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.
This report was funded by the U.S. Department of Defense.

Table of Contents

Case Study in Software Maintenance

Abstract: In an effort to find out more about the tools, procedures, and techniques project personnel use in their work, the Computer-Aided Software Engineering (CASE) Environments Project interviewed personnel in eight software maintenance projects within an agency of the U.S. government. These interviews highlighted problems that we believe are typical of many software maintenance organizations (i.e., the need for more effective software maintenance tools, lack of communication between individuals working on similar projects, low status of maintenance personnel, and lack of a design-for-maintenance philosophy during the software development phase). This report highlights the findings of these interviews, provides our analysis of the findings, and makes recommendations directed at the agency for improvement in the areas of tools, people, and process. We believe that what we observed is very typical of the state of the practice in these areas and as such that this report and its recommendations are applicable to other large or small software maintenance projects.

1 Introduction

During 1992, the Software Engineering Institute (SEI) performed a study, within a government agency, which investigated the application of computer-aided software engineering (CASE) technologies to software development. As part of this study, one task was to examine the development processes and the software tools used within maintenance (life-cycle support) projects. The examination was performed through interviews with appropriate project managers and technical personnel associated with eight projects within this agency.

This report highlights the results of these interviews. It presents our findings, recommendations, analysis of findings, suggestions for improvement, and a bibliography for further reading. The projects selected for study were chosen by agency personnel based on volunteers responding to a call for participation. The interviews were conducted over a period of three days. In most cases we spoke with the project leader and at least one other person from the project; each interview lasted approximately two hours. People spoke candidly about their work and were very enthusiastic about being given the opportunity to express both their satisfaction with and concern for the software engineering practices in their organization. In particular, they were motivated to participate by the hope that their maintenance practices and support could be improved.

While our interview sample was small relative to the size of the organization, we experienced considerable uniformity in opinions, satisfactions, and concerns across the interviews. This has led us to believe that our findings are representative of many of the software maintenance groups within the agency. Our interviews were informal; thus, we were not using any formal assessment instrument. We consider our series of interviews to be one of many steps for the agency in its drive to improve its software development and maintenance practices. This report does not provide a "silver bullet" solution for the agency: the best any such report can do is

highlight the beliefs and attitudes of the interviewees via the findings and to express our analysis and suggestions for improvement based on those findings. Improvement requires a long-term commitment by the whole organization and will involve more formal assessment and more internal analysis. The intention of this report is thus to suggest areas that need to be addressed in order to improve software development and maintenance rather than to provide a detailed improvement implementation plan.

While this study was done for a specific agency of the US government, we believe our findings have general applicability to many other government agencies and commercial companies who deal with large or small maintenance projects. Our expectations prior to the interviews were that we would focus mainly on tools and technology. However, as will become clear from reading the report, we discovered that many issues, beyond purely technological ones, hindered organizational effectiveness. We found, for example, that the agency has to confront such problems as lack of design for maintenance, low status of maintenance personnel, and ineffective communications. Communications presented a similar problem. For example, we found that diversity in project culture, compartmentalization of projects, use of different hardware platforms, etc., tended to inhibit communications. These are problems which are pervasive in many software organizations. Because maintenance activities are taking an increasingly large share of software budgets, the insights gained from studies such as this and carried into programs for managing software maintenance are of ever-increasing significance.

To protect confidentiality, the government agency involved in this study (herein referred to as "the agency") is not named. In some cases, use is made of the word "corporate". Such references have agency-wide implications. Also for reasons of confidentiality, neither the agency's specific high-level organizational groups nor the projects within these groups can be identified. The agency has a large staff which is supported by many software-intensive projects. It has customers all over the world, and has a substantial number of subcontractors. The software systems which support agency personnel are expected to be robust over long lifetimes. Two groups which were involved in the study had primary responsibility for software maintenance and software development. Thus the names "maintenance" and "development" will be used to refer to these groups or projects within these groups. One specific project which maintained a large corporate configuration management (CM) system is referred to quite extensively in the report, and is simply called "the corporate CM system."

We have organized the report as follows. Section 2 provides background information on how the interviews were conducted and describes the structure of the findings. Section 3 provides a summary and analysis that highlights the major findings and recommendations. Section 4 presents a suggested improvement plan based on our recommendations. Section 5 provides our conclusions. Appendix A contains the crux of the data we obtained in terms of our findings, recommendations for improvement, and discussions of those. Appendix B is a bibliography presenting additional information which is relevant to our recommendations.

2 The Interviews

Prior to the interviews, a group of participating candidates was identified; each participant was given a written set of topics in preparation for the interviews. The interviews were informal and covered a wide-ranging set of issues such as tools and tool use, inter-project communications, contractor and customer management, and quality assurance. The project personnel we interviewed were very candid and provided us with much valuable information. This information forms the basis of this report.

The interview results were initially structured into fifteen categories, with each category being related to a specific topic such as "Reverse Engineering and Re-Engineering Tools" or "Corporate Communications." Each of these categories contains both a set of findings and a set of recommendations related to the findings, along with a discussion of each. Details of these categories can be found in Appendix A. Over 60 findings and 70 recommendations are included.

From these, based on our analysis, three major findings/recommendations themes are derived: tools, people, and process. The tools theme deals with issues such as tools needs of maintenance projects, heterogeneous platforms, and organizational focus for tools support. The people theme deals with issues such as inter-project communication, the prestige of maintenance projects, and training of technical personnel. Finally, the process theme deals with issues such as contractor and customer interactions, schedule pressures, and quality assurance. This analysis resulted in a total of 22 major findings and 25 recommendations which are listed in Section 3. A synthesis of these findings and recommendations is given in the paragraphs below. Appendix B of this report provides a bibliography with specific references that expand on our recommendations.

Most of the information we gleaned from the interviews was generally focused on programmer activities such as coding and testing rather than on managerial activities such as budgeting and resource allocation. The interviewees did not raise issues about such activities as project management. Thus, our recommendations focus on tools, people, and process issues as they affect the work of project engineers.

The tool-related issues centered around:

- Availability, quality, and integration of CASE tools
- Importance of adequate documentation
- Improvement in configuration management support
- Lack of reverse engineering and re-engineering tools
- Difference in tooling needs for projects performing lifecycle maintenance support versus new software development
- The need for testing tools and better testing procedures

The people-related issues focused on:

- Adequacy and availability of training
- Effectiveness of communication within the agency and with customers
- Improvement in communication on the use and availability of tools
- Difficulties with contractor management
- Poor perception of maintenance activities within the agency

The process issues centered around:

- Lack of quality assurance
- Necessity to improve the software engineering standards
- Need to address improvement in the software engineering techniques and tools

Based on these findings we make specific recommendations involving tools, people, and process that we feel can be addressed by the agency. We highlight the common themes running through all the findings and emphasize the many benefits that can be accrued by addressing the recommendations through improvements in practices at the agency. We suggest developing an improvement plan and give an outline for such a plan based on our recommendations. Also, we provide a bibliography relevant to various issues that need to be addressed in an improvement plan.

3 Summary and Analysis of Findings and Recommendations

All the details of our findings, recommendations, and discussions are provided in Appendix A. They are presented as 15 categories:

- Reverse engineering and re-engineering tools
- Testing
- Configuration management issues
- Documentation
- CASE tools
- Integration
- Tooling for maintenance and new software development
- Sharing knowledge of tools
- Training of technical personnel
- Maintenance teams
- Contractor management
- Corporate culture
- Corporate communications
- Quality assurance and standards
- Hardware

All in all, there are over 60 findings and 70 recommendations. In this section, we have analyzed those findings and recommendations and found three pervasive themes. Those themes are: tools, people, and software process. Below is the distillation of all the findings and recommendations into 22 major findings and 25 major recommendations based on the three themes.

3.1 Summary of Findings

Our analysis suggests that there are common themes throughout the findings and recommendations. These themes involve tools, people, and process. The tools theme looks at technology issues related to the development of software; the people theme deals with personnel issues such as training and communication; and the process theme deals with issues such as contractor management, schedule pressures, passing on expertise, and coding philosophies. These themes are elaborated below and represent our major findings and recommendations. For each major finding, we provide a reference to the detailed discussions in Appendix A.

3.1.1 Tools-Related Findings

The major findings concerning tools and technology relate to the availability of tools to project members and the quality and maturity of CASE tools in general to assist developers and maintainers. Project members have a clear idea as to what capabilities they want from their tools but are generally disappointed by the state-of-the-art in tools. Maintainers need different tools than developers because of the legacy code they have to deal with.

The tools findings can be summarized as:

1. Most project personnel believe that they do not have adequate tools for technical support. (See Sections A.1-7.)

2. Many project personnel want additional automated support for their configuration management (CM) systems. The in-house CM system developed by the agency is in use in a number of projects, but the users believe it is only a partial solution for their needs. (See Section A.3.)

3. Projects have heterogeneous platforms which makes it difficult for the agency to identify a standard set of tools across projects. (See Section A.15.)

4. The maintenance activities which were most often cited by the projects as in need of CASE tool support include CM, reverse engineering, re-engineering, testing, and document production. (See Sections A.1-7.)

5. Most project personnel interviewed believe that they have not been exposed to state-of-the-art tools. (See SectionsA.7-9.)

6. Most new tools that have been examined by each of the projects are generally not mature enough for use by the agency. For example, we were told that many CASE tools tested are currently not sufficiently robust for maintenance of the large application systems typical of the agency. (See Sections A.1-2 and A.5-6.)

7. All project personnel believe that tooling needs for maintenance projects differ from those for new software development, and that, while tools exist for development support, tools for many maintenance activities do not currently exist within the agency. (See Section A.7.)

8. All project personnel were appreciative of the work of the in-house group responsible for evaluation, acquisition, and adoption of new CASE tools. Many project personnel told us that the work of the group should be expanded. (See Sections A.5. and A13.)

3.1.2 People-Related Findings

The major findings concerning people relate to the prestige level of maintenance work, the amount of training in methods and use of tools that people receive, and the amount of communication that occurs between personnel. The people findings can be summarized as:

1. Due to the importance of CM activities in the software maintenance process, many project personnel want to dedicate the work of one person to the role of CM support. (See Section A.3.)

2. Most people we talked to believe that within the agency maintenance is not viewed as a prestigious job. In particular, non-maintenance personnel view the CM support function as having little status. (See Sections A.3, A.10, and A.12.)

3. A number of people held the view that little rapport exists within the agency between groups dedicated to software development and those who perform maintenance activities. (See Sections A.8-10 and A.12-13.)

4. When asked whether training was adequate, a number of people told us that the right people often do not get the right technical training at the right time. (See Section A.9.)

5. A common complaint from project members is that they are not given adequate resources (i.e., time, personnel) for performing maintenance activities in a manner consistent with the agency's software quality assurance (SQA) standards. (See Section A.14.)

6. Some project personnel believe that communication within the agency is sometimes less than effective. (See Sections A.8 and A.11-13.) Specifically this involves communications between:

   - projects and upper management regarding technical needs,
   - different projects regarding the use of software tools, and
   - some projects and their customers regarding requirements.

3.1.3 Process-Related Findings

The major findings concerning process relate to the enforcement of standard practices, use of outside contractors, improvement of the working environment via new technology, and the passing on of lessons learned.

The process findings can be summarized as:

1. The majority of projects do not have documented quality assurance practices in place. In particular, systematic testing plans and test coverage analysis were carried out in only one of the projects we interviewed. (See Sections A.12-13.)

2. A number of people expressed the opinion that code is often not designed for change. Thus, while the code meets its operational specification, for maintenance purposes it is poorly designed and documented. (See Sections A.4, A.8, and A.10.)

3. A number of project personnel believe that in recent months there has been little improvement in their working environment (for example, in new software tools supporting maintenance activities). (See Sections A.10 and A.12-13.)

4. No one we interviewed knew of any agency-wide plans for introducing new technology across maintenance projects. (See Section A.11 and A.13.)

5. The use of outside contractors in some projects is seen as a cause for problems due to the lack of day-to-day control and increased communication overhead. (See Section A.14.)

6. On a number of occasions we were told that documentation creation and maintenance is one of the activities that causes a major problem. (See Sections A.4, A.7, and A.13.)

7. Many maintenance personnel we interviewed believe that the quality of the support documents accompanying the software that is passed to them is often very low. The reason most often cited for this was that schedule pressures placed on developers result in insufficient attention to the support documents. (See Sections A.12-13.)

8. A common belief amongst those interviewed is that lessons learned and expertise from previous and existing projects are not being disseminated throughout the agency. (See Sections A.4, A.7, A.10, and A.13.)

3.2 Summary of Recommendations

As with the findings, our recommendations also cover the themes of tools, people, and process. These are given in the following three subsections.

3.2.1 Tools-Related Recommendations

The tools-related recommendations can be summarized as:

1. Invest in more effective tools, particularly to support reverse engineering, re-engineering, testing, configuration management, and documentation.

2. Encourage vendors to build or adapt tools specifically to support the agency needs.

3. Improve the quality of in-house tools, such as the corporate CM system.

4. Experiment with new tools to determine their applicability to specific projects.

5. Enhance the activities of a centralized tool procurement group. Such a group might also coordinate technical communication between projects.

6. Find tool solutions that address the heterogeneous platforms needs of the agency.

7. Encourage technical personnel to more aggressively communicate their tooling needs to upper management.

3.2.2 People-Related Recommendations

The people-related recommendations can be summarized as:

1. Disseminate knowledge of the state-of-the-art in tools and expertise.
2. Assign people with particular roles (e.g., CM, testing) to projects and provide corporate support for those roles.
3. Encourage more experienced people to become maintainers.
4. Improve the prestige level of maintenance jobs.
5. Improve inter-group communications by setting up user groups, electronic bulletin boards, or vendor presentations, and encourage both the writing of in-house experience reports and inter-project meetings. Identify someone to organize and foster these communications.
6. Make training more effective, especially in tools use, standards, and documentation.
7. Establish a technical advisory board to support upper management.

3.2.3 Process-Related Recommendations

The process- related recommendations are:

1. Re-examine corporate policies in light of the increasing functionality of CASE tools.
2. Emphasize the importance of assuring that all code is as completely and accurately documented as possible.
3. Discern the difference in needs for maintenance projects versus new software development projects.
4. Support design-for-maintenance and re-engineer-for-maintenance philosophies.
5. Plan appropriate time in project schedules for documenting and testing.
6. Promote previous successes through, for example, lessons-learned reports.
7. Institute and enforce quality assurance practices.
8. Ensure that code transferred to maintenance is completely documented and meets quality assurance standards.
9. Investigate why communications within the agency are occasionally ineffective.
10. Improve customer interactions with developers and maintainers.
11. Improve contractor selection and monitoring processes in their strategic planning and decision making.

3.3 Successes

While many problems were identified by the projects, there were also positive elements which are worthy of mention. They are:

- The agency is staffed by dedicated professionals motivated by the national importance of the work they are doing.
- The skill and motivation of many the agency employees makes a critical contribution to the successful implementation, support, and use of operational systems.
- The function of the software support staff within the maintenance group is seen as crucial and valuable in technology exposure and adoption.
- The maintenance work is viewed as being challenging and exciting.

All in all, there is significant scope for improvement. The project personnel we interviewed were very keen on expressing their concerns, because in responding to our questions they saw an opportunity to initiate improvement in their working environment

4 How to Use Our Findings and Recommendations

In the previous chapter we presented an analysis of our findings and made recommendations based on those. In this section we suggest how to use our results to form an improvement plan.

4.1 Developing an Improvement Plan

Improvement takes time, planning, and resources. Based on the findings and recommendations, we therefore recommend developing an improvement plan. This suggested plan is structured into project, group, and corporate elements. The project-level issues deal primarily with the need for improved software tools. These tools are critical when addressing the immediate problem of maintaining existing poor-quality code. The group-level issues address the need for improved communications and enhanced status for maintenance personnel. Corporate-level issues focus on creating an environment in which a design-for-maintenance philosophy is seen as an integral part of the software development culture. Also at the corporate level, such issues as agency-wide quality assurance standards and contractor management are important.

All of the above issues have a relationship to Total Quality Management (TQM). The agency is currently actively involved in an extensive TQM program, and the issues addressed in this document are pertinent to that program.

To implement a successful improvement effort, several essential elements must be in place. First, visible support for the improvement plan must be publicly shown by upper management in the agency. If this does not occur, then the efforts will likely lose their momentum. Second, an action plan which identifies resources, required actions, and responsibilities must be available. Such a plan also needs grass roots input or it will not receive the agency-wide buy-in necessary for its success. Third, any improvement plan should allow for some initial short-term and easily attainable successes. If the initial goals are too ambitious, then enthusiasm may disappear under the weight of pending tasks. Finally, the plan should specify mechanisms that allow management to track the progress of the plan, in order that mid-course corrections can be made if necessary. The above elements may already be in place as a result of the TQM efforts, but they are emphasized here to highlight their importance.

4.2 Outline of an Improvement Plan

Developing an agreed-to, and implementable action plan for improvement requires a significant effort. We offer an outline that suggests a strategy for improvement. This outline represents the beginnings of a plan based on the recommendations and themes we identified earlier and is intended to meet the specific needs of the agency. The outline is given below:

1. Introduction
2. Process of Improvement
    a. improvements in a TQM context
    b. Project Needs
    c. Maintenance-Group Needs
    d. Agency Needs
3. Plan at the Project Level
    a. Tool Evaluation, Selection, and Installation
    a. Tool Training and Use
    b. Functional Needs for Tools:
        - Reverse Engineering and Re-Engineering Tools
        - Testing Tools
        - Documentation Tools
        - Configuration Management Tools
4. Plan at the Maintenance-Group Level
    a. Disseminating Knowledge
    b. Instituting a Technical Advisory Board
    c. Making Maintenance an Exciting Challenge
5. Plan at the Corporate Level
    a. Designing for Maintenance and Re-Engineering for Maintenance
    b. Improving Corporate and Customer Communications
    c. Improving Commitment to Plans and Schedules
    d. Instituting Quality Assurance Practices and Standards
    e. Managing Contractors
6. Executing and Monitoring the Plan
7. Conclusion

Section 2 of the proposed action plan identifies the major deficiencies, reviews the overall strategy to correct them, and identifies how the improvement efforts recommended in this report fit within the context of the on-going TQM initiative. Sections 3 through 5 then provide details of the plan, identifying problems, describing solutions, specifying tasks and milestones, and reviewing risk. Section 6 discusses how to implement the plan and how to monitor its effectiveness.

Developing this plan is not a short-term effort since personnel at all levels of the agency must provide input and be brought to agreement. In the plan's implementation, costs are clearly an

overriding consideration. In the long run, we firmly believe that the quality improvement actions we have described will reduce costs, but an up-front financial investment is clearly required. Thus, some form of cost-benefit analysis seems appropriate in order to establish priorities. Also, establishing a task force that is responsible for developing, implementing, and monitoring the improvement plan will help keep the project on track.

5 Summary and Conclusions

From our interviews, many problems were identified and suggestions for improvement were made by the interviewees. It is clear that the technical staff at the agency are diligent and conscientious and that they are motivated to see improvements in their working environment. We have presented our findings and recommendations in detail in Appendix A based on the information we received from the interviews. The findings and recommendations can be classified into three areas: tools, people, and process. A summary of findings and recommendations based on these themes was given in Section 3. Below, we review some of the more important findings and recommendations.

One major finding is that the maintenance group frequently has to support software which is not developed with maintainability in mind. Schedule pressures often result in software being transferred to the maintenance phase before all deliverables are completed. Consequently, the maintenance group devotes significant time to issues related to supporting software which is poorly designed, coded, tested and documented; that is, code which was not designed for maintenance. Thus we heard a consistent message that tools for such activities as reverse engineering and testing were a high priority and that several groups were not aware of the state-of-the-art in tools. We also heard that the software to be maintained does not have effective documentation to support it, either in written reports or embedded in the code. We do not believe that it is the intent of the original developers to produce incomplete work. Rather, the pressures to meet unrealistic schedules force both in-house developers and contractors to release their software before it is ready. We also saw evidence that schedule pressures within maintenance projects result in the same problem with maintained code. Thus the problem is being perpetuated in the maintenance phase. To overcome these problems, we recommend a two-phase approach. In the short term, there is a strong need for better tools to support reverse engineering, testing, configuration management, and documentation. These tools will help deal with the problems of existing code. In the longer term though, there is a need to improve the quality of the developed software. This involves such issues as developing schedules that provide more time to develop a quality product, writing effective software documentation, adhering to quality assurance standards, providing effective training on technical and non-technical issues, and coordinating communication between development and maintenance groups.

We found that communication was less effective than it could have been between different organizational entities. First, little communication appears to occur between different project personnel regarding software engineering issues. Because many of the problems confronted are common across projects, the sharing of experiences, mistakes made, lessons learned, etc., can be invaluable. Therefore communications ought to be encouraged through such means as informal reports, seminars, a software engineering bulletin board, etc. Second, we found that communication on technical issues from the project level to higher management could be improved. Some project personnel we interviewed felt that upper management was not aware of the depth of some of the technical problems they were facing. We have therefore recom-

mended that a technical advisory group be formed to keep upper management aware of the technical issues that could affect their decision making. Third, we felt that communication between projects and contractors requires reassessment. We heard that some (but not all) contractors working for the agency performed at unacceptably low levels. This could partly be due to inadequate contractor evaluation prior to contractor selection. However, it could also be due to insufficient contractor oversight during the term of the project. Clearly contractor problems can result in significant cost impact, and the use of a TQM approach to resolving this issue (e.g., working more closely with the contractor, or forming joint teams with the contractor) could be worthwhile. Finally, we heard that the effectiveness of communications with customers varies significantly. The agency policy needs to encourage customers to be involved more closely with the project throughout its life-cycle rather that just at the time of field installation.

We heard that the status and prestige of maintenance projects is less than that of development projects. Maintenance project positions are perceived as less challenging and attract personnel who have less experience. When these personnel gain experience, they move on, so that personnel stability in maintenance projects is less than it should be. However, the maintenance personnel we talked to did find their work rewarding and challenging and even preferred it to development work. Some were frustrated, though, in that they believe development groups are more likely to receive better support. To overcome these problems we feel that every opportunity should be taken to improve the status of maintenance project personnel. As stated above, they should have a measure of control over when they have to accept code into maintenance. They also need better access to the tools to do their job effectively. Finally, they should be given public recognition (inside the agency) when an effective job has been done.

The maintenance group has developed two software quality assurance documents which contain much useful information to support effective software engineering practices. If the standards in these documents were followed, then the number and severity of the product-quality problems associated with existing software development could be lessened. Unfortunately, because there has been no requirement to enforce these document standards, they often are not applied at all. We therefore recommend the following:

1. Training on quality standards should be mandatory rather than voluntary.
2. The quality standards are enforced through quality audits using an independent software quality assurance (SQA) audit function.

We also feel that the standards are deficient in several areas: they do not explicitly address issues associated with maintenance and regression testing; they do not illustrate how the standards could be tailored to the needs of projects; and, there are no examples given of how to use the standards. These aspects need to be addressed by the agency's SQA documents to be flexible enough to meet all project needs.

To address the issues in an orderly fashion in the context of improvement, three focus areas are suggested: project, maintenance-group, and corporate. The project focus is primarily on tools support, particularly for reverse engineering, re-engineering, CM, testing and documentation. These areas concern the immediate needs of the agency maintenance projects, needs

that are not being met. The maintenance-group focus is primarily on improving both effective communications within this group and improving the prestige of maintenance personnel. Finally, at the corporate level, the need is for implementing the design-for-maintenance philosophy throughout those groups which are involved in the software life-cycle.

In closing, we feel that the following benefits will result from the implementation of our recommendations:

- Improved working environment and improved morale among personnel.
- Improved quality of products through designed for maintenance.
- Improved quality products through enforcement of SQA standards.
- More effective relations with contractors and customers.
- Improved inter-project and technical/management communication.
- Long-term reduction in costs due to effective maintenance of quality software.

We feel that the problems we have brought to light in this report are indicative of the current state-of-the-art and state-of-the-practice in software engineering. Many organizations are instituting improvement plans based on recommendations similar to the ones found in this report, and significant improvements in quality and productivity are being seen. We feel sure that the agency can produce similar results by adopting the recommendations suggested in this report.

5.1 Acknowledgments

We would like to express our gratitude to agency staff for arranging the meetings and enabling us to carry out the studies. We also thank the project leaders and software engineers at the agency who took time to speak with us. Finally we would like to thank the following reviewers for their feedback: Mark Borger, Mike Caldwell, Peter Feiler, Gibbie Hart, Ed Morris, and Dennis Smith, not to mention the SEI technical writers, Dan Bidwa and Julia Deems.

Appendix A Detailed Findings and Recommendations

This appendix describes our findings and highlights a number of recommendations based on the interviews we conducted. Interviewees were encouraged to discuss in an open format the issues they were facing and their perspectives on their work. As a result their responses focussed on their current problems.

Findings have been grouped into categories representing the major areas affecting the software development and maintenance work being carried out at the agency. For each group of findings a set of recommendations is made. Similar findings and recommendations appear across some categories, highlighting dependencies between issues and illustrating a consistent pattern in the tools, people, and process needs of the agency. In general, the findings represent what we heard from the interviewees, while the recommendations are primarily ours. Sometimes, however, the distinction between a finding and a recommendation is minimal. For example, interviewees reported that no official testing plan or process existed; our recommendation is to define test plans and a testing process. Also, there were occasions when the interviewees did not explicitly mention a concern but it was easy to deduce a finding. For example, no one mentioned regression testing but discussion about other problems highlighted that the lack of regression testing was causing those problems. Thus, our finding is that no official regression testing is being carried out; our recommendation is to include regression testing as part of official development and maintenance practices.

This section is structured such that each category starts with a brief explanation of the focus of that category. Following the explanation is a list of findings accompanied by a discussion of the list. A recommendations section follows. The recommendation section begins with a description of our philosophy for addressing that category and is then followed by the actual recommendations. Finally comments on those recommendations are made.

A.1 Reverse Engineering and Re-Engineering Tools

Reverse engineering and re-engineering tools can help identify the structure and architecture of code artifacts and provide information about the codes (such as the definition and use of variables). This kind of information assists personnel in making changes to the code, restructuring it for enhancement, and making it more amenable to future change.

A.1.1 Findings

1. All life-cycle support project personnel expressed a need for reverse engineering and re-engineering tools to help analyze code under maintenance.

2. Most project personnel see tools such as CodeCenter and SMARTSystem as useful ones although a few projects found these to be immature (i.e., unstable and requiring large amounts of resources).

A.1.2 Discussion of Findings

Personnel in all projects felt that any kind of reverse engineering or re-engineering tool is an improvement over no tool. Currently projects that receive code for life-cycle maintenance generally do not receive adequate documentation. Thus, the team must rely on reverse engineering techniques in order to understand the code and to generate the required documentation. Subsequently, the team must change the code; this requires re-engineering. Teams usually have no tools to aid in reverse engineering and re-engineering. A few tools such as CodeCenter and PROCASE SMARTSystem have been evaluated. While prone to performance problems when handling large amounts of data, these tools appeared to be useful for some reverse and re-engineering tasks. Most people who had used them believed that, as these tools mature, they will be increasingly beneficial to their work.

Reverse and re-engineering tools were viewed as the primary requirement in tooling for maintenance projects.

A.1.3 Recommendations

Due to a lack of up-to-date and accurate documentation, reverse engineering tools are a vital aid in extracting information about the code in order to perform maintenance. Similarly, changes to code require change impact analysis, which some re-engineering and CM tools can provide. It should be possible to significantly reduce the need for reverse engineering tools by properly documenting all relevant information generated during the development phase. However, given that maintenance projects currently have to deal with a legacy of poor documentation, high priority should be given to providing reverse engineering tools.

Thus, our specific recommendations are:

1. Require that as much knowledge as possible about the application system is handed on from the development group to the maintenance group.
2. Investigate ways of encouraging maintainers to be involved in some parts of development activities in order for knowledge to be transferred more easily.
3. Ensure sufficient attention is paid to documentation. Do not underestimate the importance of accurate and thorough documentation.
4. As reverse engineering and re-engineering tools mature, consider investing more heavily in these tools as they are likely to be a significant asset to maintenance projects.
5. Encourage industry to build better, more suitable tools for specific platforms to aid reverse engineering and assist in re-engineering. Interactions with vendors and attendance at CASE workshops and conferences can help here.
6. Provide more support (in terms of tools, personnel, and systematic practices) for maintenance projects.in
7. Keep track of the state-of-the-art in tools such as CodeCenter and SMARTSystem, and of up-and-coming software maintenance companies.

8. Experiment with state-of-the-art tools to evaluate their applicability to agency projects. Lessons learned from these experiments should be collected and disseminated.

A.1.4 Discussion of Recommendations

One way to reduce the need for reverse and re-engineering tools is to provide the maximum amount of information and documentation about the code at the time it is passed onto the maintenance group. The ideal situation is to pass on the complete development environment to the maintenance team so that they have access to as much of the history and data about the product as possible.

When it is not possible to deliver the development environment with the operational system, all design documents should be delivered. This includes the architecture document for the product, reverse and re-engineering tools should also be provided so that the maintenance team can thoroughly analyze the code. Third-party tools to assist in this effort are starting to appear and will soon be mature. The agency needs to continue to invest in this area and experiment with various tools to see which tools are applicable to which kind of project.

Maintenance is a complex task, in large part because the documentation and other information maintenance workers receive are frequently inadequate and occasionally inaccurate. Because of this, it is a great advantage to have experienced people carrying out the tasks. Ways to encourage experienced people into the maintenance field should be investigated. In particular, it is important to train and retain people who have the enthusiasm, skills and characteristics (e.g., thoroughness, calm under pressure, inquisitive, etc.) to perform maintenance activities.

There are no tools that support defining the architecture of software applications. Having an architectural description of a software application would greatly assist reverse and re-engineering activities (along with enabling reuse of software applications). Software architectures is still a research topic, and one that the agency should definitely track.

A.2 Testing

This category concerns tools and methods for testing code.

A.2.1 Findings

1. Project personnel in most projects felt their testing procedures and testing tools were inadequate.
2. Project personnel in all projects wanted better testing tools.
3. Project personnel in many projects were not aware of the state-of-the-art in testing tools.
4. Project personnel in some projects indicated that they did not have a testing plan or strategy or regression test suite.

5. Project personnel in most projects do not have any quality control service or quality assurance person to examine the results of testing or to ensure that testing standards are being maintained.

A.2.2 Discussion of Findings

Among the project personnel we interviewed there appeared to be no common approach to the systematic testing of code. There are very few tools to aid in testing. No software quality assurance seems to be enforced to ensure that good practices are followed and that quality is built into applications. Many project personnel do not seem to be aware of state-of-the-art in testing tools, or, if they are, they do not have time to experiment with these tools.

A.2.3 Recommendations

Testing is critical to the quality of application code delivered and should figure prominently in any development or maintenance plan. In particular, a satisfactory testing plan should describe what is to be tested, how it is to be tested, and when it will be deemed to have satisfied the tests. Testing should be monitored and regression testing should be carried out after changes are made. Testing needs to be accounted for in schedules and supported with tools and practices.

Thus, our specific recommendations are:

1. Examine state-of-the-art in testing tools (e.g., TCAT, EXDIFF, T-Scope). Dedicate sufficient time to examining these tools.
2. Take a more aggressive approach to encouraging vendors to build more and better testing tools to suit agency needs.
3. Share with other projects (via testing user groups, electronic bulletin boards, distributing test plans, writing reports, etc.) information about how testing was performed.
4. Enforce the creation of better test plans, thorough testing, and SQA practices.
5. Incorporate a section on regression testing into the agency's SQA manuals.
6. Institute systematic testing (particularly regression testing) across all maintenance groups.
7. Set up independent testing groups to ensure that persons other than the programmer are also involved in testing.

A.2.4 Discussion of Recommendations

The agency needs to know about state-of-the-art in testing tools and to experiment with tools to determine which ones are appropriate to which kinds of projects. Also, SQA needs to be adopted to ensure proper testing processes are followed. A concerted effort is needed to communicate the agency's needs for testing to vendors in order to encourage vendors to build more suitable tools. Any testing expertise within the agency should be disseminated throughout the agency through seminars or in-house papers.

A.3 Configuration Management

Configuration management (CM) means different things to different people and groups. Most of the groups at the agency see CM as version control, configuration identification, status accounting, configuration control boards, and modification request tracking.

A.3.1 Findings

1. All projects are using some kind of CM. Project personnel see it as high priority and indispensable.
2. Project personnel in all projects want increased and better CM support in the sense of more automated functionality.
3. Projects have CM needs for heterogeneous platforms.
4. Many project personnel are not aware of the state-of-the-art in CM tools for their platforms.
5. Project personnel who use the corporate CM system indicated that it met their change management needs fairly well.
6. Some project personnel said they have local, specific CM needs that they feel a platform-specific CM tool should be able to meet, such as tracking derived objects.
7. Some users said they were disappointed with the reliability of the corporate CM system because of a major outage that had occurred.
8. All project personnel felt a CM person was indispensable.
9. The CM position is viewed as low-status with varied and ill-defined duties across the projects.
10. It was indicated that the corporate CM system is not a complete CM solution because it only partially meets the project CM needs. That is, there were additional needs not met by the corporate CM system.
11. Personnel in some projects do not use the corporate CM system because:
    - they did not know about it, or
    - it did not seem to address their specific requirements.
12. Project personnel felt that the corporate CM system is most suitable for version control of source objects, tracking changes and problem reporting.
13. Project personnel generally only support one release (rather than variants of releases).

A.3.2 Discussion of Findings

CM is seen as a crucial element in support of software development and maintenance. Each project uses some form of automated CM support, but most feel that more CM support is needed. For instance, while the corporate CM system was generally found suitable for version control of releases and tracking of modification requests, it lacks adequate support for code other

than source (e.g., binary and derived). For CM support that is not automated, projects are required to develop or enforce their own CM practices. Interviewees reported that on one occasion the corporate CM system proved to be unreliable (in this case, a 3 month down time and serious code losses were the result). Consequently, some users distrust the corporate CM system.

Project personnel see the CM support person as a vital part of their team. However, provisions are rarely made for a specific CM person, so projects must acquire services from outside the agency. Some projects find that they need assistance in defining the duties of their CM person in their project. (For instance, should the CM person be allowed to do builds?)

In order to simplify maintenance, generally only one release of a product is supported. While this appears to simplify development and maintenance, it can often compound the problems of system users.

A.3.3 Recommendations

Configuration management is an area that touches many aspects of a corporation including organizational level, project level, and individual programmers. It touches the corporation in the sense that change management of releases is visible at the corporate level. CM touches the project level in the sense that projects must put in place controls for managing a team of programmers as they make changes to code. CM touches the programmer via automated facilities such as version control, system modeling, and generation of derived objects. CM is a vital concern to the corporation and an invaluable mechanism for controlling the evolution of software.

Thus, our specific recommendations are:

1. Determine the corporate level, project level and programmer needs for CM and resolve which of these needs can best be met by the corporate CM system, by enhancements to the corporate CM system, and by third-party CM tools.

2. Disseminate knowledge of the state-of-the-art in CM tools in order that (a) enhancements, if necessary, can be recommended for the corporate CM system, and (b) projects can find solutions to their project specific CM needs.

3. Seek out CM automation solutions for the project-level needs on heterogeneous, distributed platforms.

4. Improve the image of the corporate CM system by making it more robust, reliable and faster.

5. Set reasonable expectations for users of the corporate CM system in terms of the kind of reliable functionality that one can expect. For example, what does the corporate CM system do and not do? (No single CM system can be a "silver bullet" for all CM needs.)

6. Re-examine the role of the corporate CM system in relation to the projects. Is its main purpose to function as an archival repository for releases and a change-tracking system for changes to releases? Is it possible to meet all the CM needs of the projects on a daily basis?

7. Re-examine the policy on single releases of applications. With more sophisticated CM support, it should be possible for programmers to develop and maintain families (variants) of product releases for customers rather than just one release.

8. Provide assistance to projects in defining more clearly the role of a CM person across the agency projects.

A.3.4 Discussion of Recommendations

In any organization, CM needs must be resolved at different levels: corporate, project, and programmer. The corporate level involves managing releases of source, tracking modification requests, and managing the resultant corporate repository. Also, many CM problems in an organization are not related to the automated CM tool but include other issues such as better training for CM library personnel and better host computers for accessing the CM tool.

Although some users believe the reliability of the corporate CM system ought to be improved, it seems to provide adequate support for the corporate level CM needs. At the project level, team members interact to create new versions of CM objects. Often a CM person is assigned to be the interface between the project and the corporate CM functions. The corporate CM system neither completely addressed the projects' CM needs nor meets the day-to-day team needs (such as communication or creation of intermediate versions of configurations). Projects need their own specific CM to support their variants, various sizes of teams and the heterogeneous platforms. A CM system specific to the platform or suitable for heterogeneous platforms would help. Then the duties of the CM person would need to include the migration of CM objects of importance from the local CM systems into the corporate system.

The developer level of CM involves those CM capabilities that are available to the individual programmer. At the very least, version control is needed. The programmer should also have a workspace in which all necessary changes, compilations, and testing can be done. Actual changes also need to be tracked; this can be difficult when the programmer is working on a machine that is not connected to the corporate CM system.

Project personnel seem unaware that CM systems can provide many more capabilities than just version control of source code. CM systems can provide versions of configurations, transactions, change request tracking, system modelling, derived object pools, and other support. However, the policies and procedures that projects use for their local CM will have to be coordinated with the corporate CM control (e.g., passing the releases into the corporate CM system repository along with details of the local change requests).

A policy is needed to achieve some consistency across projects regarding what should be put under CM control. For instance, documentation, source code, and tools should all be under CM control.

It would be worthwhile to reevaluate the responsibilities of developers, CM personnel, and the corporate CM system. Suppose, for instance, that the developers had their own CM system as part of their daily software development environment. Would the duties of the CM person change once the programmers could do their own builds under CM control (i.e., the whole generation process was recorded and tracked by the CM system)? If projects had their own CM systems, how much information would they need to pass on to the corporate CM system?

Currently, CM support is mostly carried out on source code, but for derived objects is also needed. While the corporate CM system provides a depository for such objects (binary code on magnetic tape, documentation, or floppy discs) it does not seem to correlate the source with the derived objects. A better tracking of that relationship is needed.

Because of the complexity of handling variants of releases, projects choose to support only one release of an application rather than variants customized to suit different customers. While this simplifies the software maintenance (and the CM activities), it may be an unnecessary restriction on the customers. There is a need for more automated CM support for variants, such as allowing parallel development on variants, as well as merging and parallel updates to variants. Third-party CM tools and some research CM systems provide capabilities for dealing with variants, although an automated solution is still a research issue. Such systems are worthy of examination though.

A.4 Documentation

Documentation pertains to all the relevant information about a software application that needs to be recorded. This includes requirements, design, code, tests, change logs, etc.

A.4.1 Findings

1. All project personnel were concerned about the limited amount of documentation available to them during the maintenance phase of the life cycle. In many cases, maintenance project personnel believed that they had been given incomplete or out-dated documentation when they took charge of the operational system.

2. While fixing and enhancing operational software, all project personnel had problems keeping the available documentation current (as regards the updated system).

3. There was little evidence of tool support for integrated documentation. We did hear of Software Through Pictures being used to generate design documents, while text editors were used to produce remaining documentation. Tools that captured links between code, documentation, and design diagrams did not seem to be used.

4. A few projects were experimenting with SMARTSystem to help them in the task of understanding and documenting poorly documented existing operational code.

A.4.2 Discussion of Findings

In performing maintenance activities, thorough up-to-date documentation is essential. In practice we found that such documentation did not exist for a number of projects. The reason for this is clear: pressure on development schedules leads to cutbacks on document production and maintenance. In some cases, contractor organizations had produced the code and handed it over to an agency maintenance project without sufficient documentation. While such cutbacks may benefit the development organization, the maintenance organizations often bear the legacy of such decisions.

In many cases where operational software had been in place for a number of years, the documentation was totally inadequate or out of date, and efforts were proceeding to try to reverse engineer design documentation from existing code. SMARTSystem was in use in a few projects as an aid in this task. Most project personnel expressed interest in greater tool support for these activities.

A.4.3 Recommendations

Almost invariably the maintenance of large operational systems suffered due to lack of up-to-date documentation. In practice, when a project is not meeting its schedule, the decision to suspend production of detailed documentation in favor of "all-out coding" leads to difficult maintenance problems later in the life-cycle. Personnel reported that most projects had faced this maintenance difficulty.

Our specific recommendations are:

1. Get personnel from the maintenance organizations involved early on, and continually, with the development of a system to advise on the eventual documentation needs for the system.

2. Re-examine the agency guidelines on the hand-over of operational systems from the development to the maintenance organization to assure that adequate documentation has been prepared.

3. Experiment further to examine the costs and benefits of products such as SMARTSystem which help understand and document existing operational systems.

4. Investigate the CASE tool market for other possible tools to help in the documentation and reverse engineering of existing operational systems.

5. Investigate and provide better documentation system support for developers to enable them to produce more accurate and comprehensive documentation for their systems with less human effort. Examples of technology that assists in such include structure editing systems and hypertext-based products where the relationships between code and documentation are more visible.

6. Allow time in the schedule for documentation to be created or updated.

A.4.4 Discussion of Recommendations

The recommendations fall into two broad categories. First, the process of development and hand-over of operational systems need to be improved to ensure that the needs of the maintenance organizations are more adequately met. One specific example is to encourage a member of the future maintenance project to take part in development organization meeting and provide feedback on documentation needs. Similarly, for complex systems the eventual user organization should be encouraged to have one or more personnel involved with development to ensure the end-user documentation is adequate.

Second, CASE tool support should be further investigated. Current efforts involving PRO-CASE should continue, and further studies should be initiated.

A.5 CASE Tools

Computer Aided Software Engineering (CASE) tools are third-party tools that automate parts of software development and maintenance activities.

A.5.1 Findings

1. In addition to operating system utilities, many projects were using only a small number of CASE tools to help them in their work (e.g., a design tool, CM tool, and less frequently a testing tool).
2. All project personnel thought that there were aspects of their work that might benefit from additional tool support (although they did not see the introduction of new tools as their highest priority).
3. In some cases, project personnel did not know what specific tools they required, but only the requirements they had for such tools (e.g., in the area of re- engineering).
4. Other project personnel knew of particular tools that would be of help to them, but had neither the time nor resources to bring them in-house and build up sufficient expertise to make them usable.
5. In most cases project members believed that the tools they were using were critical to the projects' success (particularly CM tools).
6. All project personnel valued the work being carried out by the maintenance group's software support staff in trying to make tools available to projects within the agency. Without this they wondered if they would have had the time and resources to obtain any CASE tools at all.
7. Few projects had any detailed knowledge of CASE tool usage in projects other than those with which they had previously been directly involved.

A.5.2 Discussion of Findings

While not viewed as being as important to success as having qualified personnel and a well defined development process, all projects recognized the importance of CASE tools support. Most often the tools that personnel cited as critical to their work were IDE's Software through

Pictures for software design, and PROCASE's SMARTSystem for CM. Additionally, a number of projects have been investigating the use of SRI's Software TestWorks to help them in their testing activities. The corporate CM system, developed within the agency, was also used in a number of projects for problem tracking and reporting.

A strong reason for the use of these particular tools appears to be the influence of a CASE tool coordination role performed by the software support staff, which provided many of these CASE tools (at no cost) to the projects we interviewed.

In discussions about the benefits of introducing more CASE tools, most project personnel believed that there were aspects of their work that could be significantly improved with additional tools. Two areas were most often cited as prime candidates for additional tools:

1. Testing. Most projects had a mainly ad hoc approach to generating test cases, documenting test results, and feeding results to the other stages of the life cycle. It was felt that tool support for these tasks would be a significant benefit.

2. Reverse Engineering. A number of projects had significant problems with maintaining code for which design and rationale documentation were non- existent, out of date, or of poor quality. The ability to take a running system and reverse engineer some parts of this documentation would greatly facilitate future maintenance of these systems.

Some projects had been investigating tools that helped in these tasks, but little day-to-day support currently exists in these areas.

A.5.3 Recommendations

The introduction and use of further CASE tools offers much potential benefit to the projects we interviewed. While it is clear that tools in themselves will not solve many of the development and maintenance problems experienced by the projects, additional tool support can have an impact in some areas.

It is interesting to note that the project members themselves recognized where additional tool support would be of benefit, and where their manual procedures were adequate. This points to the fact that the introduction of a wide-ranging, comprehensive, automated support facility would be counterproductive in some cases. A much more flexible approach to gradually and incrementally expanding a project's tool set seems appropriate.

Little inter-project communication occurs regarding the tools being used in different projects. By improving inter-project communication, information can be shared and reused. CASE tools tend to be large, complex artifacts that require significant effort to build up sufficient expertise to make their use of practical benefit. This steep learning curve is much more costly if each project experiences it in isolation. While the work of the software support staff has had an important impact in providing a central pool of knowledge on CASE tools, there are many more tasks that can help support improvement in this area.

Our specific recommendations are:

1. Raise awareness of the range of CASE tools currently available in the marketplace (e.g., by inviting vendors to make presentations at the agency).

2. Build up an accessible body of knowledge on CASE tools for use within the agency so that tool users have access to information on the availability, use, costs, and benefits of CASE tools.

3. Be more systematic in the reuse of existing conventions and tool know-how across projects by capturing lessons learned from CASE tool use, documenting naming conventions and usage scenarios for tools, and developing guidelines that facilitate their use.

4. Provide greater encouragement and promotion of CASE tool user groups; this perhaps requires the agency resources and management commitment to ensure they are attended and successful (e.g., paying for external speakers).

5. Expand the role of a central group for CASE tool acquisition, promotion, and introduction as currently performed by the software support staff.

6. Continue dialog with CASE tool vendors to ensure that the vendors are aware of the agency's particular needs with regard to CASE tool support.

A.5.4 Discussion of Recommendations

Many of the problems identified in our discussions can be addressed by increasing the visibility of CASE tool use and practices within the agency. While work is taking place in this area, it is important to highlight the main activities involved. Three key aspects need to be recognized.

1. Greater publicity, recognition, and information transfer from the successful application of CASE tools is required. There do seem to be users of CASE tools who have important positive experiences with their use. There are also less successful examples of CASE tool use. It is important that information about such experiences are as widely disseminated as possible.

2. Having a central point of contact for the acquisition, promotion, and encouragement of CASE tool use does seem to be having a positive effect. While issues of funding, resource requirements, and penetration will continue to be a problem, this work should be expanded.

3. Management support and recognition for these activities will be essential to its success. Providing the time and resources to find out about new CASE tools, build up expertise, and consistently and effectively use the tools will always be a thorny issue.

A.6 CASE Tool Integration

We here address integration of CASE tools in support of the development process.

A.6.1 Findings

1. Obtaining or assembling an integrated set of CASE tools was viewed as less important than the need to have skilled project personnel communicating and interacting effectively.

2. Use of integrated tool sets, while expressed as desirable, was rarely a high priority for the projects.

3. In most cases the integration of CASE tools took place through manual data transfer or simple automated work-arounds.

4. Most CASE tools were being introduced in a piecemeal fashion as particular needs arose, or as opportunities to acquire particular tools came along.

A.6.2 Discussion of Findings

To most project personnel interviewed, personnel issues were the major contributor affecting the productivity and quality of their work. CASE tools were seen as important, but only because they empowered project personnel. As a result, CASE tools had not been acquired and adopted as part of a well-defined procedure to create an integrated tool set.

For a project to operate effectively, integration, cooperation, and coordination needs to occur on (at least) three levels:

1. Between project personnel. A working environment is required that facilitates and encourages sharing and cooperation between project personnel. Physical location, for example, is one factor affecting this. The fact that one project was moved from being in close proximity on a single floor to being dispersed on a number of floors was highlighted as having a significant detrimental effect on the progress of the project's work.

2. Between CASE tools. Most projects used tools acquired at different times and from different vendors. As a result, they had to devise manual procedures and conventions, and simple automated techniques for using the tools together.

3. Between project personnel, CASE tools, and development process. Personnel and tools must work together to implement an effective development or maintenance process. Most projects found it to be a struggle to ensure that each of the three elements were supporting the other two, rather than at the expense of the others.

A.6.3 Recommendations

While clearly such issues of integration are complex and difficult to address, the effects of even small improvements would have far-reaching consequences on project effectiveness. In particular, it seems unlikely that the introduction of more complex automated support will have significant impact unless it also supports changes in the development approach to take advantage of those tools.

Our specific recommendations are:

1. Promote an approach to CASE tool integration that considers at integration to be a design activity that ties the use of collections of CASE tools to the particular needs of project personnel in support of an effective process. This should provide a better context in which to evaluate potential CASE tool purchases.

2. Continue experimentation and analysis of integrated tool sets so as to determine their possible usefulness to the agency's projects. This will help in increasing understanding of the different approaches to CASE tool integration, and in enumerating the benefits and costs associated with each one.

3. Facilitate communication of lessons learned in both the CASE tool integration experiments and the current use of CASE tools in the agency's projects.

4. Pay particular attention to the CASE tool needs of long-term maintenance projects, as opposed to new development projects. Dealing with code that has a legacy requires a focus on different techniques such as reverse engineering and re-engineering.

5. Encourage projects to define a tool integration strategy document during project start-up, which will address tool integration needs in supporting the project's development and maintenance process.

A.6.4 Discussion of Recommendations

The usefulness of integrated tool sets in supporting the agency's projects requires further experimentation and evaluation to provide greater insight into the current state-of-the-practice. There are frequent announcements of new "integrated CASE tools" and "integration products." These should continue to be studied carefully with regard to their possible use within the agency. Long-term maintenance of operational systems poses very stringent requirements on CASE tools. Most existing tools do not meet these requirements.

Additionally, project personnel should be encouraged early in a project's life to describe the anticipated tool needs of the project and to create a plan for acquiring and using tools in support of the project. Two important aspects of such a document are the integration strategy proposed for the tools and an analysis of the flexibility of such an approach in allowing the tool set to grow and evolve over the lifetime of the project.

A.7 Tooling for Maintenance and for New Software Development

This concerns the relationship between tool needs in a maintenance setting as opposed to those in a recently-started software development project.

A.7.1 Findings

1. Maintenance projects and new development projects have different priorities and needs.

2. Maintenance personnel indicated that they have inadequate tooling to support them.

A.7.2 Discussion of Findings

There is an impression that projects doing new software development are given more support (in terms of personnel and tools) than maintenance projects. New projects have no legacies to deal with since they are starting from scratch; maintenance projects however, have to contend with whatever quality of product is given to them and whatever artifacts (such as documentation) are missing. Tools are needed to assist the maintenance team.

A.7.3 Recommendations

While in theory new software development and maintenance should be similar activities and thus require similar tools, this is not the case in practice. Code handed over to maintenance is often accompanied by insufficient and out-dated documentation. The maintainer then has to deal with the legacy of the operational system over extended periods of time. Thus, there is a different emphasis placed on the importance of particular tools for new software development compared to maintenance. In our recommendations we take the viewpoint of a maintenance engineer. Our specific recommendations are:

1. Discern the differences in emphasis between maintenance versus new projects and provide appropriate tooling and practices for each.

2. Invest in tooling for maintenance in the following order of importance:

    a. reverse engineering (such as capturing the structure of the code)

    b. re-engineering (such as assisting in restructuring the code)

    c. testing (such as collecting unit tests for the regression suite)

    d. configuration management (such as control over the derivation of executable objects)

    e. design (such as drawing the architecture of the code)

    f. documentation tools (such as capabilities for easily placing pictures into text)

    g. integration of tools (such as Software Through Pictures with SMARTSystem with code generation capabilities and examining integration frameworks such as HP's Softbench or Sun's ToolTalk)

    h. metrics tools (such as data for management that will aid in resource allocation and scheduling)

3. In the adoption of maintenance tools, we suggest the following set of activities:

    a. define the life-cycle process model

    b. acquire a set of CASE tools that support design, documentation, configuration management, testing, debugging and simulation (if necessary)

    c. integrate tools in a way that encourages data interoperability between the tools

    d. provide training on tools and the life-cycle model for programmers

    e. educate the team on the whole view of the project

    f. provide team building exercises

4. Deliver the development environment with the code to aid maintenance. If that is not possible, provide for interoperability between the development and maintenance environments.

5. Support a design-for-maintenance philosophy for new projects; that is, by developing quality software and support materials in order to avoid unnecessary problems faced during the life-cycle support phase. There is no easy solution for this since it requires a corporate commitment to improved software design.

A.7.4 Discussion of Recommendations

The tooling needs for maintenance are based on the need to analyze code, change code, test the changes, and keep a history of those changes. The design-for-maintenance philosophy emphasizes the need to consider the maintenance needs of an operational system during development (e.g., extensive documentation and testing). In this way, maintenance activities are less likely to be a major problem in the future. Studies have shown that the expense of fixing problems early in the life-cycle is many times less than fixing the same problem once the system is in operation.

A.8 Sharing Knowledge of Tools

Sharing knowledge about the functionality and availability of tools is the topic of this category.

A.8.1 Findings

1. Upper management does not seem aware of the tooling needs of technical personnel.

2. There is a lack of coordination in tools procurement throughout the maintenance group.

3. Lessons learned about specific tools are not disseminated.

A.8.2 Discussion of Findings

We were told that upper levels of management do not seem aware of the tooling needs of life-cycle support projects. This, coupled to the perceived low status of maintenance projects, results, at least in some cases, in too little funding for software tools. However, we also heard that some management does try to support tool needs, but that funding is simply stretched too thin.

Having central CASE tool information available has been much appreciated by the project personnel we interviewed. However, there seemed to be a lack of coordination between the groups themselves in terms of their tools needs. If more coordination occurred, it is likely that greater leverage could be applied to the tool vendors when tools are purchased.

Cultural isolation of projects within the agency is widespread. This isolation is often unnecessary and tends to prevent valuable experience from being disseminated between projects.

A.8.3 Recommendations

Technical personnel must inform management of their tool needs. Management must have a plan for the introduction of tools. Thus, our specific recommendations are:

1. Encourage technical personnel to more aggressively market their project tools needs to upper management.
2. Coordinate the purchase of tools beyond the project level.
3. Foster communication between projects. Means of doing this include: initiating user groups for specific tool areas, holding seminars on tool issues and products, organizing an electronic bulletin board, and producing short reports on tool experience.

A.8.4 Discussion of Recommendations

It may be helpful for personnel on different projects to meet to identify common types of tools used, and to develop a plan for organizing tool purchases. Such a plan would provide upper management with a broader picture of the maintenance support needs of the maintenance group (e.g., why design, reverse engineering, and testing tools are essential components). The plan could form the basis for interaction with vendors and provide a consistent basis for such interaction across maintenance-group projects.

With the pressure of getting the job done, it is easy to ignore the above communication issues as they appear to consume valuable time. However, in the longer run, communication on tool needs and experiences will more than pay off in terms of productivity and quality.

A.9 Training of Technical Personnel

Training concerns the kind of education provided to personnel developing and maintaining code.

A.9.1 Findings

1. Training is not sufficiently encouraged since time is not clearly allocated for this function.
2. Pressure to get the job done is a factor in inhibiting training. We were told that the right training was not given to the right people at the right time.
3. Some technical staff are not provided with a knowledge of the products they are working on.
4. Because of turnover of in-house contractor personnel, training costs for these individuals was high.

A.9.2 Discussion of findings

While training facilities do exist, the evidence suggests that they appear to be inadequately used. For example, training is given on SQA standards, but this training is provided on a voluntary basis. Because of tight schedules, few personnel appear to participate. Further, there does not appear to be a systematic approach to training personnel; for example, training is not built into project schedules, nor does management vigorously support the training of technical personnel. It was expressed that the personnel most likely to receive training were those who had the time rather than those who had the need, and that people "feel guilty" about taking time off for training.

It was expressed to us that programming staff sometimes do not understand the wider context of the components they are working on. In fact, one view expressed was that it might be desirable for programmers to perform their assignments in isolation from related activities.

It takes time for new contractor personnel to become effective and integrated members of any project. They are likely to lack an understanding of the organization in which they find themselves or the project's history, of the tools and techniques which the project uses, and of the standards which apply. With significant turnover in contractor personnel, the implicit costs of contractor education can be high.

A.9.3 Recommendations

Training is a fundamental task for any corporation. A corporate plan must be established and enforced. The right people must receive the right training at the right time. Our specific recommendations are:

1. Plan for incorporating more time and resources into project schedules in order to encourage training. Training costs should be a visible part of project costs.
2. Ensure that training policies and plans are visibly supported by management and that training results are tracked to ensure effectiveness. Coordinate vendor training on software products (e.g., have training classes on tools involving personnel in multiple projects). This could be less expensive than ad hoc vendor training.

3. Enforce internal training on the agency software standards. (See also section on Quality Assurance and Standards.)

4. Provide informal training to the programming staff about the nature of the project and products they are working on.

5. Make efforts to retain qualified contractor personnel or to rehire those who, because of prior the agency experience, will require less training.

A.9.4 Discussion of Recommendations

Training needs to be visibly supported by management, and training policies and plans should be understood by all. "Visible support" implies that management at and above the project level take a positive public stand on promoting training. Certain training should be mandatory (e.g., on standards); other training should be highly encouraged (e.g., in documentation production). Further training may be enforced depending on the context (e.g., if a tool is to support a project, then the engineers in questions should have training in the tool's use.)

The up-front costs of training are clear; this is a disincentive to investing in an effective training program. However, while the costs of not training are less tangible, they are often higher in the long run. Training needs to focus on issues which reduce costs and improve quality in the long term, such as training for effective documentation (not just writing, but the whole documentation process) and writing of well-designed code with meaningful comments. Such improvements can have a dramatic effect on life-cycle maintenance costs. Second, coordinated training will bring technical personnel into greater contact with each other, and better communication will result from uniform practices (e.g., in performing software inspections). Finally, by coordinating all internal and external training, training consistency will improve.

A.10 Maintenance Teams

This concerns the personnel who make up the maintenance teams and the difficult task of performing maintenance.

A.10.1 Findings

1. Many personnel believed that maintenance activities have low prestige, are poorly supported by management, and have a low priority at the corporate level.

2. Project personnel felt that not enough experienced people are assigned to maintenance activities.

3. Code being turned over to maintenance generally does not come with adequate documentation.

4. Code turned over to maintenance is generally not of high quality.

5. Many maintenance personnel saw their jobs as important, generally exciting and challenging. However, this view did not seem to be shared by non-maintenance personnel.

6. Code is generally not designed to be amenable to change.

A.10.2 Discussion of Findings

The task of performing maintenance is generally held in low esteem. However, the people who choose to carry out maintenance see it as a most challenging task, particularly in the context of variable quality code, missing supporting documentation, and lack of consideration for maintenance needs during design.

A.10.3 Recommendations

Maintenance is a very challenging job and requires motivated, clever people. These people need to be supported in terms of tools and rewards. Thus, our specific recommendations are:

1. Improve the prestige level of maintenance projects by providing more and better tools, hiring more experienced personnel and increasing training effectiveness. Include rewards and enticements.

2. Designate specific roles (such as CM person, test manager, and documentation manager) in maintenance teams and recognize the importance of these roles (for example, through rewards and titles). For large projects, these roles can be filled by a person. For small projects, one person may hold multiple roles. Roles need to be recognized and supported by management since they are crucial to maintenance and support an attitude of quality in process and product.

3. Set up a user group to capture and promote more effective ways of performing maintenance and sharing solutions and work-arounds.

A.10.4 Discussion of Recommendations

Maintenance is a difficult and challenging job. Teams performing maintenance require motivated and experienced personnel. These individuals need to be dutifully supported and rewarded.

The quality of applications passed on to the maintenance team needs to be higher. The institutionalization of quality assurance practices will assist in such, as will providing better tools that capture all the information about the application. Extra time should be included in schedules to allow for the development and updating of documentation.

Much expertise is ignored when applications are passed on to maintenance. It is necessary to capture that expertise and pass it on. This can be done by including maintenance people in the development team or vice versa. Also, user groups should be set up to meet a few times each year just to pass on information about lessons learned during development and maintenance.

A.11 Contractor Management

Contractor management concerns the important relationship that exists between contractor personnel and the agency.

A.11.1 Findings

1. Lack of supervision on external contractors has often resulted in reduced product quality.
2. Because of high turnover, training costs for in-house contractor personnel are high.

A.11.2 Discussion of Findings

We were told that the quality of contracting organizations varied widely. This variation in quality was such that, at least as reflected in some of the project personnel we interviewed, major in-house rework had to be performed. This poor quality could be a symptom of inadequate contractor evaluations during source selection process.

The cost of inadequate contractor work is quite high. Lack of supervision of contractor organizations results in the delivery of products which are often poorly documented and not well designed for maintenance. This problem is exacerbated by the pressures imposed on contractors to meet schedules, with the result that some major problems are identified only after delivery.

An important point to note is that the turnover of in-house contracting personnel results in high training costs since new personnel often need to be trained in the techniques used by the project. A hidden cost is incurred since new personnel take some time to understand their work in the context of the project.

A.11.3 Recommendations

Evaluating and monitoring contractors is a difficult and time-consuming task. While we recognize that ensuring the quality and correctness of contractors' work is in many cases impossible, we make the following specific recommendations that may improve the situation:

1. Re-examine the contractor evaluation process prior to awarding the contract. For example, a technique such as SEI's Software Capability Evaluation (SCE) could be used to lower the risk associated with source selection.
2. Form joint customer/contractor teams to work on mutual problems.
3. Improve the monitoring of contracting personnel and contracted products. As in 1) above, SCE can be used to support this monitoring process. In the monitoring phase, SCE's can be carried out by support or contractor personnel.
4. Reward contractors who demonstrate quality improvements with:
   - reduced supervision
   - additional contract awards

5. Encourage contractors to train their personnel in needed disciplines.

6. Retain or rehire those contract personnel for in-house work who have already been trained and/or have familiarity with the agency's culture.

A.11.4 Discussion of Recommendations

As has often been stated, going with a contractor simply because it offers the lowest bid may turn out to be a costly error. However, it is recognized that such a practice as favoring certain suppliers may conflict with current agency (or DoD) regulations.

While we did not investigate the cause of poor contractors, inadequate contractor evaluation during source selection is a strong possibility. If this is the case, use of a technique such as SEI's Software Capability Evaluation could be considered since it provides a formal and consistent way of evaluating multiple contractors.

Improved monitoring of the contractor will ensure that costly mistakes are less likely to occur. The degree to which monitoring is required depends on the past performance of the contractor and the degree to which the contractor is familiar with the technical and management issues of the project.

A.12 Corporate Culture

Corporate culture concerns the kind of culture that exists within the agency. We particularly focus on the ways in which projects interact, common perceptions within groups, and beliefs about how improvement in the working environment is attained.

A.12.1 Findings

1. Project groups and their work tend to be too compartmentalized; this reduces both the amount of communication and information sharing.

2. Improvement in tools, practices, and techniques is difficult because very little technical information and experiences are shared and expertise is not pooled (centralized).

3. Maintenance does not have as prestigious a reputation within the organization as does new software development. There appears to be a "them versus us" attitude between developers and maintainers which raises the level of distrust and disharmony.

A.12.2 Discussion of Findings

Many interviewees felt there was little improvement in their working environment over time with regards to expertise being disseminated, better tools being introduced, and better quality applications being built and maintained.

A.12.3 Recommendations

A corporate culture is produced by all the people who work in the corporation and is promulgated by management. To change the culture requires strong and active leadership.

Thus, our specific recommendations are:

1. Promote and disseminate successful solutions to technical problems within the agency and encourage others to follow suit.
2. Investigate why there are so many limitations to the sharing of information between projects and pooling of expertise.
3. Enhance the status of maintenance organizations through:
   - encouraging greater interplay between related development and maintenance groups
   - empowering maintenance groups with control over acceptance of newly developed software
   - providing incentives for personnel to take on maintenance responsibilities

A.12.4 Discussion of Recommendations

Expertise needs to be valued by being rewarded. This can be done in many ways. For example, direct, personal awards can be made to individuals. More indirect recognition can come from encouraging individuals to disseminate their expertise through seminars or papers, or being made consultants to other projects.

Corporate-wide policies for improving quality and expertise, such as via TQM practices, need to be adopted. This requires adequate training time and the adoption of good quality tools. The status of existing software engineering environments needs to be thoroughly evaluated and a plan put in place to ensure that the quality of the environments will progress over time. An improvement philosophy needs to be adopted. People need to see that improvement is a priority and that it will happen.

A.13 Corporate Communications

Corporate communications concern how a corporation communicates with its own employees, advocates communication between projects, and how it shares and disseminates the technical expertise of its members.

A.13.1 Findings

1. Effective communication, whether between software maintainers and contractors or between software maintainers and customers, varies widely.
2. In some cases, technical personnel believe that management is not sufficiently aware of the technical needs of their projects.

3. Impediments to a coordinated project team included the use of outside contractors for some or all of the design and implementation work (due to cultural, educational and motivational differences) and dispersed physical location of project personnel (which increases communication problems).

A.13.2 Discussion of Findings

In some cases, interactions with customers are very close. In one case, a customer representative stayed with a project for a period of years; in another, the customer was part of a project's change control board. However, in another case, scheduling time to meet with a customer proved difficult; this resulted in requirements not being well specified. Lack of communications with contractors (e.g., lack of effective contractor reviews) has resulted in some serious problems. For example, software has had to be totally rewritten as a result of inadequate contractor work. Communication between projects, for example, in software engineering issues, was very limited. Thus, valuable lessons learned were not exploited.

A.13.3 Recommendations

Communication between projects and people should be encouraged since this is the best way to share expertise and resolve issues. Thus, our specific recommendations are:

1. Establish a corporate policy that encourages frequent meetings between the maintenance projects and their customers and contractors.
2. Encourage customers to interact with project developers during the earlier phases of the project, not just during the product installation phase.
3. Encourage inter-project communications in technical areas. This could involve, for example: holding seminars, encouraging active special interest groups, establishing bulletin boards (if possible), and documenting experiences through internal papers.
4. Involve senior technical personnel in upper-management meetings and decision-making that involves technical issues. One suggestion is to establish a technical advisory board to keep management abreast of current technical issues and problems.

A.13.4 Discussion of Recommendations

More frequent meetings between maintenance personnel and the customer will help identify problems (e.g., in requirements) early on, when they are more easily corrected. Regular review of contractor work, as it is being performed, will reduce the likelihood of major undiscovered problems. Delays in identifying such problems have, in the past, resulted in significant cost increases.

Many project personnel we talked to who were facing similar technical problems in, for example, CM and software engineering tools support. We found that the software support staff was much appreciated because of its efforts to support the technical needs of the projects. However, direct project-to-project interaction seemed to be lacking. By fostering communications between projects, not only are costs likely to decrease due to improved coordination of tech-

nical needs, but lessons learned can be more widely disseminated. As a result, similar problems can be handled more effectively. The work taking place to establish an "experience factory" at NASA's Goddard Space Center may be a useful model to examine. (A reference to this work is given in Appendix B.4.)

In some cases we heard that upper management was supportive of projects with respect to both tools and hardware. In other cases, we heard the upper management "do not understand the real needs of the development level," and that projects had to do too much "scrounging around." It would thus appear that, in some cases, better communications between upper management and technical personnel would be beneficial. Hence we make the final recommendation to foster communications between technical staff and management.

A.14 Quality Assurance and Standards

Quality assurance concerns following a process that produces a high quality product. Usually standards are designed to encourage such a process.

A.14.1 Findings

1. We found that the agency has, in general, a well-defined set of SQA standards for the practice of software engineering, although these do not address some important areas in detail (e.g., software maintenance).

2. While most technical personnel appear to be aware of the agency SQA standards, few projects appear to be enforcing them.

3. While training on standards is provided, few projects take advantage of this training since it is provided on a voluntary basis.

4. There does not appear to be any independent SQA organization or activity which assures that quality-related standards are being met.

A.14.2 Discussion of Findings

The agency's SQA standards appear to form a stable basis from which to ensure effective software quality. However, in no projects where personnel were interviewed are SQA standards being enforced in all appropriate areas. Enforcement of these standards should be at the agency organizational level, rather than only at the maintenance-group level. This is important since the maintenance group does not work in a vacuum, but has to interact with other developers (for example, in the development group), with customers, and with contractors. If none of these groups adopt the SQA standards, then it is hard to see how maintenance group can effectively apply them.

We found that the agency standards did not cover certain issues which should have been covered. In particular, the standards did not cover maintenance activities. In addition there was no explicit mention of regression testing. This form of testing is particularly important to a maintenance organization.

We heard that, relative to software development, maintenance is a low-status activity. This perception is likely to have a long-term negative impact on software quality. We were told, for example, that maintenance groups had too little control over the quality of software being delivered to them. This results in a disincentive for development groups to produce a quality product, particularly with respect to product documentation. We also heard that the maintenance group tends to attract relatively inexperienced personnel (a symptom of the low status of the work), and that there is a problem in keeping young software engineers in the maintenance area. Maintenance is a complex activity requiring a diverse set of skills. The challenges and rewards of performing maintenance tasks need to be emphasized.

A.14.3 Recommendations

To ensure quality in process and product, practices such as those of Total Quality Management (TQM) should be in place along with supporting corporation standards and policies. Thus, our specific recommendations are:

1. Provide training in SQA standards to all personnel.
2. Put in place a mechanism to ensure that the agency's SQA standards are being followed throughout all the agency groups involved in software engineering.
3. Arrange for life-cycle support personnel to monitor and provide guidance to developers during development efforts in order that products have maintainable characteristics.
4. Enforce a policy that products are not to be accepted for life-cycle support until they have passed a set of quality tests.
5. Enhance the agency's SQA standards with:
   - regression testing practices
   - a focus on maintenance as well as new software development
   - examples of how to use the standards
   - tailored guidelines so that projects can adapt the standard's practices to meet their specific needs
6. Review the agency's SQA standards for additional items in light of the current agency emphasis on TQM.

A.14.4 Discussion of Recommendations

In order to ensure compliance with the SQA standards, enforced training in these standards must be given. However, training itself is insufficient. Independent auditing of projects must also be carried to ensure compliance. Therefore, an SQA organization or activity, with an independent (non-project) reporting channel should be established. The objective of this function is primarily to guarantee projects are following the procedures in the standards, rather than directly checking the quality of the products themselves.

The maintenance group must be empowered to control the quality of the products delivered to them. With enforced quality standards in place, the need for the maintenance group to reject incoming products should be significantly reduced. The ability of the maintenance group to control its receivables will also improve if the problem of the group's perceived low status is addressed.

A.15 Hardware

Hardware concerns the kinds of platforms used and the implications for CASE tool support.

A.15.1 Findings

1. The project personnel interviewed employed a wide variety of hardware platforms, and at times experienced considerable difficulties because of this heterogeneous environment.
2. The impact of this wide variety of hardware was that skills, knowledge, and CASE tools were often not reusable from one project to another.
3. One group felt that they were not receiving sufficient operating system and CASE tools support due to employing a less common hardware and system software configuration.

A.15.2 Discussion of Findings

The age and long life of many operational systems as well as the diversity of the agency customers have led to a wide variety of hardware platforms being used. This introduces obvious difficulties in terms of the skills required to use the hardware, and the lack of easily transferable CASE tools from one hardware platform to another. Reuse of people, skills, and support software such as CASE tools is severely limited in such an environment.

One of the consequences of a working environment where heterogeneous systems are used is that people working on less popular hardware feel as though they are out of the mainstream and are receiving insufficient attention. For example, most project personnel we interviewed had Unix-based workstations as their development environment, and the major CASE tools that were used operated in this environment. The existing VMS-based projects are not able to transfer much of this knowledge to their working environment.

A.15.3 Recommendations

While diversity of hardware platforms is inevitable, careful consideration must be taken to try to provide CASE tool support that is as portable as possible across as many different hardware configurations as possible. Issues of which hardware platforms are supported by a CASE tool should be determined as early as possible.

Our specific recommendations are:

1. Support heterogeneous platforms. When selecting and acquiring CASE tools, particular care should be taken in to considering the needs of developers using many different hardware platforms. CASE tool support is required on all platforms.
2. Insist on a requirement of portability. In discussions with CASE tool vendors, the agency should make questions of tool portability across multiple hardware platforms a high priority.
3. Continue investigations in the area of hardware and platform standardization efforts (e.g., POSIX) and their potential influence on the agency.

A.15.4 Discussion of Recommendations

With the speed at which hardware improvements are taking place, it seems clear that there will not be the opportunity to have a single hardware platform at any foreseeable point in the future. In this situation, CASE tool vendors should be asked in detail about their current and future plans with regard to supporting multiple platforms, client/server architectures, support for existing standards, and so on.

Appendix B Recommended Reading

Below is a list of reports and articles that we feel are most appropriate for the issues confronting the agency. They are grouped by topic. A brief annotation is provided for each.

B.1 Addressing Quality

[Mansir 89] Mansir, B.E.; & Schacht, N.R. "Total Quality Management: A Guide to Implementation." Maryland: Logistics Management Institute, August 1989.

- Presents a detailed discussion on Total Quality Management and its strategy for implementation.

[Brown 91] Brown, P. G. "QFD: Echoing the Voice of the Customer." *AT&T Technical Journal* (March/April 1991): 18-32.

- Describes an approach that allows the customer to have an effect on the development of the product.

[Caldiera 92] Caldiera, G.; & Cantone, G. "A Reference Architecture for the Component Factory." *ACM Transactions on Software Engineering and Methodology 1*, January 1992.

- Discusses the concept of the "experience factory" which relates to improved inter-project communication and capturing lessons learned.

B.2 Addressing Risk

[Boehm 89] Boehm, B.W. *Software Risk Management.* California: IEEE Computer Society Press, 1989.

- Discusses many issues of risk management.

[Charette 89] Charette, R.N. *Software Engineering Risk Analysis and Management.* Intertext Publications / McGraw-Hill Book Company, 1989.

- Looks specifically at the risk issues affecting software.

[Kirkpatrick 92] Kirkpatrick, R. J.; Walker, J.A.; & Firth, R. "Risk Management: An SEI Appraisal" *SEI Technical Review '92*, Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

- Gives a description of risks associated with software development.

[Fairley 89] Fairley, R.E. "Risk Management: The Key to Successful Software Projects." *Proceedings of Third IFAC Workshop on Management of Software Projects.* Indiana: Purdue University: Oct. 30 - Nov. 1, 1989.

- Describes a systematic approach to risk management.

B.3 Addressing Technology Transition

[Pressman 90] Pressman, R.S. "Managing the Transition to a Software Engineering Environment." Software Engineering: Tools, Practice, Auerbach Publishers, Vol. 1, No. 2, July/Aug. 1990: 33-41.

- Details technology transition to a software development environment.

[Fowler 92] Fowler, P.; & Levine, L. "Toward a Defined Process of Software Technology Transition." *American Programmer* March 1992.

- Presents models for technology transition.

[Bouldin 89] Bouldin, B. *Agents of Change: Managing the Introduction of Automated Tools.* Englewood Cliffs, NJ,: Yourdon Press, 1989.

- Discusses the issues of introducing CASE tools into an organization.

[ODR 85] O. D. Resources Inc. "Building Commitment to Technological Change." Georgia: O.D. Resources Inc. 1985: 7.

- Highlights a phased approach toward instituting change.

B.4 Addressing Process Improvement

[Humphrey 88] Humphrey, W. S. "Characterizing the Software Process: A Maturity Framework." *IEEE Software*: March 1988: 73-79.

- Describes five levels of process maturity within an organization.

[Humphrey 91] Humphrey, W. S.; Snyder, T.R.; & Willis, R.R. "Software Process Improvement at Hughes Aircraft." *IEEE Software*: July 1991: 11-23.

- Presents an example of improving the maturity of a corporation.

[Grady 87] Grady, R.B.; & Caswell, D.L. *Software Metrics: Implementing a Company-Wide Program.* Englewood Cliffs N.J.: Prentice Hall, 1987.

- Explains capturing metrics in an organization.

B.5 CASE Tools

[Zarrella 90] Zarrella, P. *CASE Tool Integration and Standardization* (CMU/SEI-90-TR-14, ADA235640). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, Dec. 1990.

- Gives a short discussion of a number of standardization efforts in relation to CASE tool integration.

[Huff 92] Huff, C.; Smith, D.; Morris, E.; & Zarrella, P. *Proceedings of the CASE Management Workshop* (CMU/SEI-92-TR-6, ADA258234). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

- Presents the results of a workshop dedicated to CASE acquisition issues.

[Holbrook 91] Holbrook, H.B.; & Jones, L.G. *An International Survey on Experiences with Software Maintenance Tools* The Hague: Shape Technical Centre Consultant Report 78, April 1991.

- Gives a survey of software maintenance tool usage.

[Morris 91] Morris, E.; Feiler, P.; & Smith, D. *Case Studies in Environment Integration* (CMU/SEI-91-TR-13, ADA248152). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, Dec. 1991.

- Presents results of studying several environments and examining what issues affect integration.

[IEEE 92] IEEE Computer Society *Proceedings of the 5th International Workshop on Computer-Aided Software Engineering*, Montreal, Canada: IEEE Computer Society Press, July 1992.

- Contains papers presented at a workshop dedicated to CASE tools and their use.

B.6 Configuration Management Issues

[Dart 91] Dart, S.A. "Concepts in Configuration Management Systems," pp 1-12. *Proceedings of Third International Workshop on Software Configuration Management.* Trondheim Norway: June 1991.

- Gives a description of the state of the art in CM systems.

[Dart 92] Dart, S.A. "The Past, Present and Future of Configuration Management," *Proceedings of IFIP World Congress.* Spain: Sept. 1992.

- Gives a review of issues facing practitioners, managers, and vendors for the future of CM systems.

[Feiler 91] Feiler, P. H. *Configuration Management Models in Commercial Environments* (CMU/SEI-91-TR-7, ADA235782). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, March 1991.

- Presents a categorization of CM systems for programmers.

[Whitgift 91] Whitgift, D. *Software Configuration Management: Methods and Tools.* London: John Wiley and Sons, June, 1991.

- Gives an overview of CM concepts and a number of CM systems.

[SCM] *Proceedings of International Workshops on Software Configuration Management,* Los Alamitos, CA: ACM SIGSOFT, IEEE Computer Society (4 workshops held).

- Presents a series of workshops on current research in CM issues.

B.7 CASE Tool Integration

[Wallnau 92] Wallnau, K.C. *Issues and Techniques of CASE Integration with Configuration Management* (CMU/SEI-92-TR-5, ADA253323). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, March 1992.

- Presents models for viewing integration of CASE and CM tools.

[Brown 92] Brown, A.W.; Feiler, P.H.; & Wallnau, K.C. "Understanding Integration in a Software Development Environment." *Proceedings of the 2nd International Conference on Systems Integration.* Morristown, NJ: IEEE Computer Society Press, June 1992.

- Discusses a services-based view of CASE tool integration.

[Brown 92a] Brown, A.W.; Feiler, P.H.; & Wallnau, K.C. "Past and Future Models of CASE Integration." *Proceedings of the 5th International Workshop on Computer-Aided Software Engineering*: Montreal, Canada: IEEE Computer Society Press, July 1992.

- Gives a review of CASE integration approaches.

[Thomas 92] Thomas, M.I.; & Nejmeh, B. "Definitions of Tool Integration for Environments." *IEEE Software 9,* 2, March 1992, 29-35.

- Presents a set of definitions and goals for CASE tool integration.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-93-TR-08	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-93-185
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (city, state, and zip code) HQ ESC/ENS 5 Eglin Street Hanscom AFB, MA 01731-2116
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESC/ENS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO: 63756E; PROJECT NO.: N/A; TASK NO: N/A; WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) A Case Study in Software Maintenance	
12. PERSONAL AUTHOR(S)	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM TO	14. DATE OF REPORT (year, month, day) July 1993 15. PAGE COUNT 50 pp.
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
FIELD	GROUP	SUB. GR.	CASE tools environments computer-aided software software maintenance engineering environments

19. ABSTRACT (continue on reverse if necessary and identify by block number)

Abstract: In an effort to find out more about the tools, procedures, and techniques project personnel use in their work, the Computer-Aided Software Engineering(CASE) Environments Project interviewed personnel in eight software maintenance projects within an agency ot the U.S/ government. These interviews highlighted problems that we believe are typical of many software maintenance organizations (i.e., the need for more effective software maintenance tools, lack of communication between individuals working on similar projects, low status of maintenance personnel, and lack of a design-for-maintenance philosophy during the software development phase). This report highlights

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution	
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas R. Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631	22c. OFFICE SYMBOL ESC/ENS (SEI)

ABSTRACT — continued from page one, block 19

the findings of these interviews, provides our analysis of the findings, and makes recommendations directed at the agency for improvement in the areas of tools, people, and process. We believe that what we observed is very typical of the state of the practice in these areas and as such that this report and its recommendations are applicable to other large or small software maintenance projects.